CMPUT 331, Fall 2023, Assignment 5

All assignment submissions must conform to the Assignment Submission Specifications *posted on eClass. Ensure that your submission follows these specifications before submitting your work.*

You will produce a total of four files for this assignment: "a5p1.py" for problem 1, "a5p2.py" for problem 2, and a README file (also in either PDF or plain text).

All encipherment and decipherment in this assignment uses the simple substitution cipher. Lowercase and uppercase letters are to be treated as the same.

Problem 1

Create a Python module named "a5p1.py", containing functions named "freqDict(ciphertext)" and "freqDecrypt(mapping, ciphertext)". When invoked with a text enciphered using the substitution cipher, the "freqDict(text)" function should perform frequency analysis on the entire text, using the frequency statistics methods from the textbook, and return a dictionary whose keys are the cipher characters, with the value for each key being the plaintext character it is assigned to using frequency analysis. For English texts that have been deciphered using this *frequency analysis* method, the value of the most frequent cipher character should be 'E', and the value of the least frequent cipher character should be 'Z'. If two or more letters occur the same number of times in the ciphertext, the letters that occur earlier in the alphabet should be considered to have a higher frequency. The "freqDecrypt(mapping, ciphertext)" function simply accepts a dictionary mapping of characters (in the format returned by "freqDict") and returns the resulting text after applying the mapping to each character in the ciphertext.

Note: This method of cracking the substitution cipher is far from perfect and it is unlikely to return the correct key.

We provide the letters in the alphabet sorted by the frequency in English (from Wikipedia's article on Letter Frequency):

```
ETAOIN = "ETAOINSHRDLCUMWFGYPBVKJXQZ"


>> freqDict("AABBA")['B']
"T"

>> freqDict("-: AB CD AH")['A']
"E"

>> freqDict("MKLAKAALK")
{'A': 'E', 'K': 'T', 'L': 'A', 'M': 'O'}

>> freqDecrypt({"A":"E","Z":"L","T":"H","F":"O","U":"W","I":"R","Q":"D"},
                "TAZZF UFIZQ!")
HELLO WORLD!
```

Problem 2

There are two primary ways of evaluating a solution to a simple substitution cipher: *key accuracy* and *decipherment accuracy.* Key accuracy is the proportion of cipher character types in the alphabet which are mapped to their correct plaintext characters. Decipherment accuracy is the proportion of cipher character tokens in the ciphertext which are mapped to their correct plaintext characters.

Create a Python module named "a5p2.py" that contains a function named "evalDecipherment(text1, text2)" where text1 is a plaintext and text2 is an attempted decipherment of a ciphertext created by enciphering text1. The evalDecipherment function should compare the two files, and return a list containing two fields that correspond to the key accuracy and decipherment accuracy of text2 w.r.t the plaintext, text1. For example, if text1 contains the plaintext "this is an example", text2 might contain "tsih ih an ezample" – the text in text1 was enciphered, and text2 contains an imperfect attempt to decipher it. This decipherment has a key accuracy of 8/11, since there are 11 character types, and three of them, 'h', 's', and 'x', were deciphered incorrectly; it also has a decipherment accuracy of 11/15, since it is 15 characters long, but only 11 character tokens in the decipherment are correct. Thus, the function should return the list [0.7272727272727273, 0.7333333333333333]. Your program should only count alphabetical characters in its decipherment evaluation and it should be case-insensitive.

```
>> evalDecipherment("this is an example", "tsih ih an ezample")
[0.7272727272727273, 0.7333333333333333]

>> evalDecipherment("the most beautiful course is 331!",
        "tpq munt bqautiful cuurnq in 331!")
[0.7142857142857143, 0.625]
```